有道移动广告API接口规范

Version 1.0.7

2016-07-13

概述

对于希望接入有道移动广告投放的开发者而言，使用移动广告SDK（Android/iOS/JavaScript）是推荐的接入方式。不能嵌入SDK的开发者，可通过广告API的方式接入，这种方法较SDK有更多开发工作量。

本文档面向以广告API方式接入有道移动广告的移动开发者。

产品要求

1. 目前有道移动广告提供了Native广告，近期将提供Banner/插屏等更多样式的广告。
2. 由于移动终端的网速原因，为了给用户更好的体验，建议开发者仅在3G/4G/WIFI下请求大素材广告。
3. 从用户体验和转化角度出发，建议广告点击后处理逻辑如下:
4. 广告主投放的普通链接，iOS用外置浏览器打开，Android直接用内置浏览器打开。
5. 广告主投放的iOS应用，直接跳转到APP store。
6. 广告主投放的Android 应用，非wifi下弹出二次确认框，需要用户确认后进行下载，wifi下直接下载。
7. 一次请求可以请求多条广告，开发者确保在广告真实展示或点击的情况下分别触发展示请求或点击请求。
8. APP客户端向有道广告服务器发送请求，不能以APP开发者服务器向有道广告服务器发送请求。
9. 开发者需要提供开发完成的安装包给有道移动广告方进行测试，经过测试审核通过后方能把新版应用发布到应用商店。

开发准备工作

1. 在有道开发者系统平台注册“应用”并创建“广告位”。
2. 根据API协议文档接口说明方法向有道移动广告服务器发送广告请求，并处理广告响应和数据上报。

开发总体原则

1. 通过HTTP方式与广告服务器端进行通信

2. 不论GET还是POST方式，参数值以UTF-8编码。如果是 POST 方式，请使用 application/x-www-form-urlencoded 作为 Content-Type，POST 目前不支持 JSON 格式的数据。
3. 根据广告请求广告类型不同，返回广告内容部分取值略有不同。当X-Adtype为html时，返回内容是一个符合html5规范的html片段。开发者应将该html片段通过webview等方式渲染给用户。html片段中可能包含嵌入式js/css或文件引用式js/css，展示容器必须能够执行、解释相关js/css。不允许修改html片段中的任何字符。当X-Adtype为json时，返回内容为广告原始数据，需要开发者进行展示结果渲染，然后展示给用户。
4. 通过API请求到的广告内容，请及时展现给终端用户，不要缓存，不要同一广告展现多次，这样广告服务会将该点击判为作弊，反而使得开发者收入降低。
5. 开发者在广告请求时应尽可能多填写可选参数，这样广告服务可以配到更相关、用户更喜欢的广告，能够提升用户点击率，从而提高开发者收入。但是如果开发者本身获取不到可选参数，请开发者不要随便填写可选字段，这样反而可能造成配到不相关广告，降低点击率，影响开发者收入。总之，对于可选字段，开发者应该尽最大可能填写准确可信的值。
6. 请求广告时，若从服务器发起，必须使用 rip 参数传入用户原始 IP 地址。
7. 展示上报和点击上报必须从客户端发起，禁止通过第三方服务器发起，以免被判定为作弊。

接口说明

请求

请求地址

http://gorgon.youdao.com/gorgon/request.s

请求方式

GET或POST

请求参数

Field	Scope	DataType	Description
id	required	String	移动广告位ID
ct	required	Integer	网络连接类型，（UNKNOWN, ETHERNET, WIFI, MOBILE;），值可能为0,1,2,3，分别以上几种广告连接状态
dct	required	Integer	子网络连接类型。当ct字段为MOBILE时添加，为3G时值为12，4G时为13
udid	required	String	设备ID,如AndroidID或IDFA
imei	required	String	IMEI（International Mobile Equipment Identity）是移动设备国际身份码的缩写. IMEI 与

			udid 至少传入一个。
ll	recommended	String	位置信息，GPS或者网络位置，经纬度逗号分隔（经度在前，纬度在后）
s	recommended	String	加密请求时使用此字段，其值为整个加密后的字段
ydet	recommended	Integer	加密请求时使用此字段，表明具体使用的加密方法
dn	optional	String	设备信息，如samsung,GT-S5830,GT-S5830
z	optional	String	时区，如：+0800
o	optional	String	竖屏横屏,可能值分别为:p,l,s,u; u:未知,p:portrait,l:landscape,s:square
sc_a	optional	String	屏幕分辨率，值如：1.0
mcc	optional	String	国家类型，如中国460
mnc	optional	String	运营商，如移动00联通01
iso	optional	String	国家代号，值如cn
cn	optional	String	运营商名，值可能为‘中国联通’
lac	optional	String	小区码
cid	optional	String	基站码，更加准确定位位置
wifi	optional	String	wifi信息，用户将当前连接或者附近的wifi的ssid和mac传送过来，非当前连接无法获取mac，格式上，第一个参数为当前wifi的mac，第二个为当前wifi的ssid，第三个以后就是其他网络的mac和ssid，参数以逗号分隔，如：ac:f7:f3:a4:bc:5a，NetEase，ac:f7:f3:a4:bc:5a，liaofan，ac:f7:f3:a4:bc:5a，outfoxer
ran	optional	Integer	一次请求的广告数量，默认值为1。如果大于1为批量广告请求（请注意广告返回格式），建议一次不要请求太多广告，如果有需求可以分批取
cids	optional	List	在同一个信息流广告会话中，会将最新加载的广告推广创意的ID都传送给服务端，以便服务端进行广告去重。示例：cids=1,2,3。那么本次请求将不会返回变体id为1, 或者 2 或者 3 的广告
rip	optional	String	用户原始 IP 地址，当通过服务器请求广告时，必须传入此参数
isrd	optional	String	当希望访问 clktracker 后跳转到 clk，即通过点击跟踪链接由有道的服务跳转到落地页时，可以通过 isrd=1 来启用此参数。如果启用此参数，无

			须通过访问 clk 进入落地页。

广告请求加密

为了保证API请求的安全，推荐使用加密方式请求广告。目前API提供了明文、BASE64、DES加密的方式请求广告。如果采用DES加密方式，API使用者需要使用注册广告位时生成的广告加密密钥来加密。采用DES加密方式请求广告时，将所有请求参数(广告位ID除外)加密后放入Field(s)字段并加上加密方式参数Field(ydet)。使用BASE64编码需要将广告位ID放入Field(s)字段。

注意：各参数的值应该首先使用URLEncoder.encode(value)进行编码。

Field(ydet)参数说明:

Value	Description
0	无加密，此时可省略此参数
1	使用广告位对应的密钥DES加密后BASE64编码
2	字符串BASE64编码后反转

以下是各加密方式对应的java版本例子。

DES:

```
public static byte[] encrypt(String str) throws NoSuchAlgorithmException,
    NoSuchPaddingException, InvalidKeyException,
    IllegalBlockSizeException, BadPaddingException,
    InvalidKeySpecException, UnsupportedEncodingException {
  SecureRandom sr = new SecureRandom();
  SecretKey key = getPrivetKey();
  Cipher cipher = Cipher.getInstance("DES");
  cipher.init(Cipher.ENCRYPT_MODE, key, sr);
  byte datas[] = str.getBytes("utf-8");
  byte[] encryptedData = cipher.doFinal(datas);
  return encryptedData;
}
```

BASE64:

```
String base64 = Base64.encodeToString(bts, Base64.DEFAULT);
return StringUtils.reverse(base64);
```

广告请求样例(Android)

明文

```
http://gorgon.youdao.com/gorgon/request.s?
id=8ccf454bdcf051daa19ef1bce1b10fd2&ll=39.993987,116.322652&dn=Xiaomi,MI
2,aries&udid=sha:79a8397d32d91cb25bce87fce98567e41ff4b950&z=+0800&o=p&sc\_a=
2.0&ct=2&dct=-1 &wifi=ac:f7:f3:18:fd:fc,"NetEase",aa:b2:89:66:9d:ce,outfoxer
&imei=860308023497320
```

DES加密

```
http://gorgon.youdao.com/gorgon/request.s?
id=8ccf454bdcf051daa19ef1bce1b10fd2&s=XxDt00UG3kzHg1LU8s3V3U9f8xR0oHafTqkoCE
OQyNTXxo%2BoRxcJpjjAgKuViCqvxf8boGqlOa9n6GdkHWDzSrE0W8X8tJLM8dhIJr39UWalfZgV
L8Inw2LpwML8bFsDNP%2BUWAC8aYxKniApSLOD1yy%2BQch&ydet=1
```

BASE64编码

```
http://gorgon.youdao.com/gorgon/request.s?
s=aWQ9YXNkZmFzZGZhZGZhc2RmYQ==&ydet=2
```

请求响应

服务端的HTTP返回状态码一律为200，广告请求的响应消息在http response header中，广告内容在http response content中。不同广告样式请求具有不同的返回头信息，返回结果数据格式以http header中的X-adType为准，目前提供的native广告返回结果是json格式，具体参数如下：

Native AD Response Header

Field	Description	Value sample
X-Adtype	广告返回类型	html，json
X-Clickthrough	点击地址	http://p.clkservice.youdao.com/clk/request.s?k=...
X-Height	广告高度，针对banner和插屏广告	350
X-Width	广告宽度，针对banner和插屏广告	240
X-Imptracker	广告跟踪	http://dsp-impr2.youdao.com/k.gif?yd_ewp=7727&...
X-Launchpage	landpage	http://www.youdao.com
X-Creativeid	推广创意ID，长整型	1505140

Native AD Response Content

广告请求返回的Content信息即为广告内容，Native AD的广告内容由广告变体决定，分为必有字段和可选字段。必有字段为有道平台制定的字段，可选字段为媒体自定义字段。

Field	Scope	DataType	Description
creativeid	required	Integer	广告 id
clk	required	String	点击链接
~~clktracker~~	~~required~~	~~String~~	~~点击跟踪链接，~~**已失效，请使用clktrackers，过渡期内该字段将保留**
clktrackers	required	Array	点击跟踪链接数组
imptracker	required	Array	展示跟踪链接数组
ydAdType	required	Integer	广告类型，0：落地页广告；1：下载类型广告
iconimage	optional	String	
mainimage	optional	String	
text	optinal	String	
title	optional	String	

例如：

```
{
  "text":"新增口语训练 ",    //文本
  "title":"有道词典",    //标题

"iconimage":"https://d30x8mtr3hjnzo.cloudfront.net/creatives/5ab7121a5a4247d
6b17cc921235ae28b",    //小图

"mainimage":"https://d30x8mtr3hjnzo.cloudfront.net/creatives/0eaee6486b7c4e0
4a38d2cae365995dd",    //主图
  "ydAdType":0,
  "clk":"http://www.youdao.com",
  "clktracker":"http://ads.youdao.com/clktracker0"
  "clktrackers":
["http://ads.youdao.com/clktracker0","http://ads.youdao.com/clktracker1","ht
tp://ads.youdao.com/clktracker2"],
  "imptracker":
[http://ads.youdao.com/imptracker0","http://ads.youdao.com/imptracker1","htt
p://ads.youdao.com/imptracker2"]
}
```

Multiple Native AD Response Content

批量广告请求的返回广告内容与单个广告请求基本相同，只是返回的Response Content是Json Array，且每个Json Object中包括creativeid属性。

例如:

```
[
  {
    "title":"有道词典",    //标题
    "text":"新增口语训练 ",    //文本

"iconimage":"https://d30x8mtr3hjnzo.cloudfront.net/creatives/5ab7121a5a4247d
6b17cc921235ae28b",    //小图

"mainimage":"https://d30x8mtr3hjnzo.cloudfront.net/creatives/0eaee6486b7c4e0
4a38d2cae365995dd",    //主图
    "creativeid":"324234",    //推广创意ID
    "clk":"http://www.youdao.com",
    "clktracker":"http://ads.youdao.com/clktracker0"
    "clktrackers":
["http://ads.youdao.com/clktracker0","http://ads.youdao.com/clktracker1","ht
tp://ads.youdao.com/clktracker2"],
    "imptracker":
[http://ads.youdao.com/imptracker0","http://ads.youdao.com/imptracker1","htt
p://ads.youdao.com/imptracker2"]
  }
 ]
```

展示上报

广告展示给用户时，开发者需要实时将本次展示事件上报给有道移动广告系统，如果没有进行展示事件上报或展示上报延迟很多，广告系统可能会将此次展示产生的点击判为作弊流量，将会影响开发者收入。

展示上报URL为广告相应内容中的imptracker字段。开发者需要在广告展示时向imptracker中的所有url发送HTTP请求。

展示上报中响应头中status code为200且返回1个像素点表示上报成功,否则失败。当有大量上报失败的情况时请及时向有道广告平台商务人员反馈。

点击上报

用户点击给广告时，开发者需要实时将本次点击事件上报给有道移动广告系统，如果没有进行点击事件上报或上报延迟很多，广告系统可能会将此次点击判为作弊流量，将会影响开发者收入。

点击上报URL为广告相应内容中的clktrackers字段。开发者需要在广告点击时向clktrackers中的所有url发送HTTP请求。请务必保证点击上报时，请求中的 User Agent 不为空。

点击上报中响应头中status code为200且返回“thanks”表示上报成功,否则失败。当有大量上报失败的情况时请及时向有道广告平台商务人员反馈。

注意

为便于应用快速审核通过，Android的load、展示、点击请求建议使用HttpClient而非HttpURLConnection实现。

附录

请求参数获取示例（Android）

ct参数获得:

首先通过

```
int type = activeNetworkInfo.getType();
```

获取系统网络类型，然后调用下面转化方法得到ct的值

```
switch (type) {
  case TYPE_ETHERNET:
    return ETHERNET;
  case ConnectivityManager.TYPE_WIFI:
    return WIFI;
  case ConnectivityManager.TYPE_MOBILE:
  case ConnectivityManager.TYPE_MOBILE_DUN:
  case ConnectivityManager.TYPE_MOBILE_HIPRI:
  case ConnectivityManager.TYPE_MOBILE_MMS:
  case ConnectivityManager.TYPE_MOBILE_SUPL:
    return MOBILE;
  default:
    return UNKNOWN;
}
```

其中：*UNKNOWN* (0), *ETHERNET* (1), *WIFI* (2), *MOBILE* (3);

dct参数获得:

```
dct默认值为-1
if(网络类型ct为MOBILE即为3)
  dct = activeNetworkInfo.getSubtype();
}
```

udid参数获得：

```
String deviceId = Settings.Secure.getString(
      context.getContentResolver(), Settings.Secure.ANDROID_ID);
    deviceId = (deviceId == null) ? "" : Utils.sha1(deviceId);
return "sha:" + deviceId;
```

其中：

```
public static String sha1(String string) {
        StringBuilder stringBuilder = new StringBuilder();
        try {
            MessageDigest digest = MessageDigest.getInstance("SHA-1");
            byte[] bytes = string.getBytes("UTF-8");
            digest.update(bytes, 0, bytes.length);
            bytes = digest.digest();
            for (final byte b : bytes) {
                stringBuilder.append(String.format("%02X", b));
            }
            return stringBuilder.toString().toLowerCase();
        } catch (Exception e) {
            return "";
        }
   }
```

imei参数获得：

```
public String getImei() {
    try {
      final TelephonyManager telephonyManager = (TelephonyManager) mContext
          .getSystemService(Context.TELEPHONY_SERVICE);
      if (telephonyManager != null) {
        return telephonyManager.getDeviceId();
      }
    } catch (SecurityException e) {
    }
    return "";
}
```

dn参数获得：

```
mDeviceManufacturer = Build.MANUFACTURER;
mDeviceModel = Build.MODEL;
mDeviceProduct = Build.PRODUCT;
```

包含这三个值，以,分割

mcc,mnc 参数获得

```
protected void setMccCode(String networkOperator) {
    String mcc = networkOperator == null ? "" : networkOperator.substring(
        0, mncPortionLength(networkOperator));
        addParam("mcc", mcc);
}

protected void setMncCode(String networkOperator) {
    String mnc = networkOperator == null ? "" : networkOperator
        .substring(mncPortionLength(networkOperator));
        addParam("mnc", mnc);
}

protected String getNetworkOperator() {
    String networkOperator = mTelephonyManager.getNetworkOperator();
    if (mTelephonyManager.getPhoneType() == TelephonyManager.PHONE_TYPE_CDMA
        && mTelephonyManager.getSimState() ==
TelephonyManager.SIM_STATE_READY) {
        networkOperator = mTelephonyManager.getSimOperator();
    }
    return networkOperator;
}

private int mncPortionLength(String networkOperator) {
    return Math.min(3, networkOperator.length());
}

protected void addParam(String key, String value) {
    if (value == null || isEmpty(value)) {
        return;
    }

    mStringBuilder.append(getParamDelimiter());
    mStringBuilder.append(key);
    mStringBuilder.append("=");
    mStringBuilder.append(Uri.encode(value));
}
```

Android请求加密：

```
public static String encryptUrl(String url) {
    int s = url.indexOf("?");
    if (s > 0) {
      String paramString = url.substring(s + 1);
      String encrypt = encryptUrlParams(paramString);
      return url.substring(0, s + 1) + "s=" + encrypt;
    }
    return url;
}

public static String encryptUrlParams(String params) {
    String s = params;
    try {
      byte[] encrypts = EncryptUtil.encrypt(params);
      s = Uri.encode(EncryptUtil.getBase64(encrypts)) + "&ydet=1";
    } catch (Exception e) {
      try {
        s = Uri.encode(EncryptUtil.getReverseBase64(s.getBytes("utf-8")))
            + "&ydet=2";
      } catch (UnsupportedEncodingException e1) {
        return Uri.encode(s);
      }
    }
    return s;
}

public static byte[] encrypt(String str) throws NoSuchAlgorithmException,
      NoSuchPaddingException, InvalidKeyException,
      IllegalBlockSizeException, BadPaddingException,
      InvalidKeySpecException, UnsupportedEncodingException {
    SecureRandom sr = new SecureRandom();
    SecretKey key = getPrivetKey();
    Cipher cipher = Cipher.getInstance("DES");
    cipher.init(Cipher.ENCRYPT_MODE, key, sr);
    byte datas[] = str.getBytes("utf-8");
    byte[] encryptedData = cipher.doFinal(datas);
    return encryptedData;
}

private static SecretKey getPrivetKey() throws InvalidKeyException,
      InvalidKeySpecException, NoSuchAlgorithmException {
    DESKeySpec dks = new DESKeySpec("广告位对应的密钥".getBytes());
    return SecretKeyFactory.getInstance("DES").generateSecret(dks);
}
```